Swapping as the "*Hello World*" of Relational Reasoning: Replications, Reflections and Extensions

Donna Teague and Malcolm Corney
Faculty of Science and Technology
Queensland University of Technology,
Brisbane, QLD, Australia
{d.teague,m.corney}@qut.edu.au

Alireza Ahadi and Raymond Lister
Faculty of Engineering and Information Technology,
University of Technology, Sydney,
Sydney, NSW, Australia
Raymond.Lister@uts.edu.au

Abstract
At the previous conference in this series, Corney, Lister and Teague presented research results showing relationships between code writing, code tracing and code explaining, from as early as week 3 of semester. We concluded that the problems some students face in learning to program start very early in the semester. In this paper we report on our replication of that experiment, at two institutions, where one is the same as the original institution. In some cases, we did not find the same relationship between explaining code and writing code, but we believe this was because our teachers discussed the code in lectures between the two tests. Apart from that exception, our replication results at both institutions are consistent with our original study.

Keywords: Novice programmer, tracing, explaining, writing.

1 Introduction
A number of recent research results have demonstrated a relationship between the ability of novice programmers to manually execute ("desk check" or "trace") code, their ability to explain the purpose of a piece of code, and their ability to write similar code. Lopez et al. (2008) found that, when tracing and explaining were each used separately in a single regression model, neither tracing code nor explaining code were strong indicators of code writing ability. However, when combined in a multiple regression, tracing code and explaining code accounted for 46% of the variance in marks awarded to a code writing question in an exam. Venables, Tan and Lister (2009) performed a similar study, and also found a strong relationship between tracing, explaining and writing, as illustrated in Figure 1.

Lister, Fidge and Teague (2009) also studied the relationship between tracing, explaining and writing, but they used a non-parametric approach. As part of their results, they effectively screened students on their code tracing ability, which allowed them to isolate and study the relationship between code explaining and code writing for a sample of students with a tracing performance >50%. For those students, Lister, Fidge and Teague found that students with ≤50% score on code explaining tasks performed statistically worse on a code writing task than students who scored >50% on the code explaining tasks (see Table 1). Similar results have since been reported for students at other educational institutions (Lister et al., 2010). From these studies, it seems plausible (but not proven) that a student is ill prepared to write code if that student also does not have reliable code tracing skills, or code explanation skills. If asked to design and write code, such a student may have little alternative but to engage in programming by "random mutation", as the student may lack the analytic skills necessary to systematically debug their own code.

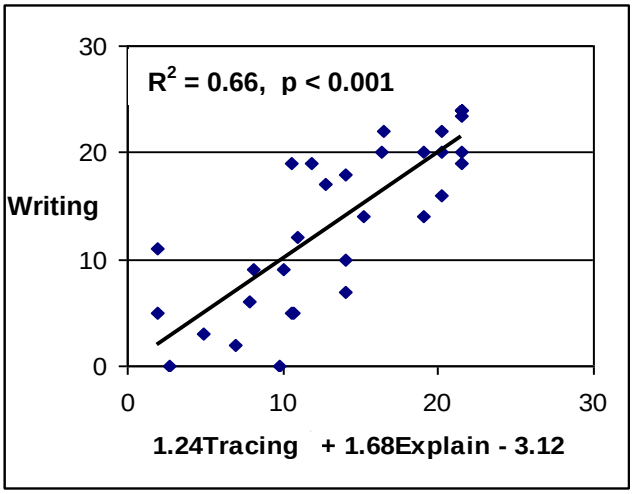

Figure 1: A multiple regression, from Venables, Tan and Lister (2009), with score on code writing as the dependent variable, and the combination of scores on tracing and explaining as the independent variables.

Number of good answers on four explanation questions	Success on a code writing question
> 50% (n = 98)	67%
≤ 50% (n = 24)	46%
χ^2 test	p = 0.05

Table 1: Empirical results from Lister, Fidge and Teague (2009), showing the relationship they found between code explaining and code writing. (This table is derived from Table 7 of their paper.)

1.1 Corney, Lister and Teague (2011)

The above empirical studies all collected data from students at the end of their first semester of learning to program. In earlier work (Corney, Lister and Teague, 2011), we tested a class of CS1 students at three points in their development – at week 3, again at week 5, and at the end of semester. One of the questions in the week 3 test required students to answer the code explanation question shown in Figure 2.

> The purpose of the following three lines of code is to swap the values in variables `a` and `b`:
>
> ```
> c = a
> a = b
> b = c
> ```
>
> The three lines of code below are the same as the lines above, but in a different order:
>
> ```
> a = b
> b = c
> c = a
> ```
>
> In one sentence that you should write in the box below, describe the purpose of those second set of three lines. **NOTE:** Tell us what the second set of three lines of code do all by themselves. Do NOT think of those second three lines as being executed after the first three lines of code.
>
> *Sample answer*: "it swaps the values in b and c"

Figure 2: A question from the week 3 test of Corney, Lister and Teague (2011).

In the week 5 test, one of the questions required students to write code to swap the values of two variables (i.e. code similar to that shown in Figure 2). After eliminating from that sample those students who had performed poorly on some code tracing tasks, we found that students who had successfully explained the above swapping code in week 3 were much more likely to write correct code for swapping two variables in week 5 than the students who could not explain the code in week 3. In addition, the students who performed better on these questions in week 3 and week 5 performed better on a code writing task at the end of the semester. These results led us to conclude that the problems some students face in learning to program begin very early in semester.

1.2 Overview

In this paper, we present replications of our earlier work (Corney, Lister and Teague, 2011), performed at two institutions. One of the institutions is the Queensland University of Technology (QUT), which was the source of the data in the original study. The other institution is the University of Technology, Sydney (UTS). The QUT replication is very similar to the original study, while the UTS replication contains some variations from the original study.

2 Replication at QUT

In the replication at QUT, students were tested at week 3 and week 5 of semester, the same weeks as in the original study. As in the original study, these students were learning Python. In both weeks, we used the same test questions as in our original study. Also, we screened for and eliminated novices, using the same tracing questions as in our original study. After that screening, 51 students remained in our sample.

2.1 Writing a Swap

Table 2 summarises the results from this replication. The percentages shown in the brackets (and preceded by "cf.") are the percentages from our original study. Our replication results do not support the results in our original study. The most notable difference in our data is that a far higher percentage of our students who could not explain a swap at week 3 could write a swap at week 5 (i.e. 71% cf. 57%).

Week 3, Explain a swap between two variables (see Figure 2)	Week 5, Successfully wrote an equivalent swap between two variables
Wrong (n = 21)	71% (cf. 57%)
Right (n = 30)	83% (cf. 92%)
χ^2 test	p = 0.3 (cf. p = 0.001)

Table 2: Results from the replication at QUT, with comparative percentages shown in brackets from our original study (Corney, Lister and Teague, 2011).

Week 3, Explain a swap between two variables (see Figure 2)	Week 5, Successfully wrote a swap between two variables	
	failure	success
Wrong (n = 21)	6	15 (i.e. 71% of 21)
Right (n = 30)	5	25 (i.e. 83% of 30)

Table 3: The contingency table for calculating the chi-square value in Table 2.

2.1.1 Reflections

We suspect that the difference in our results is due to the way in which these tests were integrated with our teaching. In the original study, the week 3 test was not discussed with the class by the lecturer (i.e. co-author Corney). In contrast, our week 3 test was followed by a lengthy discussion of the test by the lecturer. (There was nothing pedagogically novel about that discussion. Corney discussed swapping in the same way a lecturer might discuss any piece of code.) Assuming our explanation for the difference is correct, our result may be encouraging, as it may indicate that the problems students face are amenable to pedagogical intervention. However, the question would still remain as to whether a student can transfer that learning to other programming problems. This is an issue to which we return in section 2.2.2.

In Table 2, we have presented our data in the same format we used in our original work. That format is an unusual format for presenting data that is then tested statistically by a chi-square test. Table 3 reproduces our data from Table 2 as the more traditional contingency table. In this paper, we will present our results in both forms.

2.2 Explaining a Sort of Three Variables

Another question in the week 5 test from our original study is shown in Figure 3. In that study, we reported a statistically significant result ($p < 0.05$) for student performance on this question in week 5 and the explanation question in week 3. Our replication results are shown in Tables 4 and 5. While our percentages in the replication are very similar to the percentages in our original study, our replication results do not quite meet the traditional 0.05 threshold of statistical significance, perhaps due to our smaller sample size.

Opinions vary on the interpretation of the 0.05 threshold for statistical significance. Some people view it as a rigid threshold − a result is either significant (i.e. below 0.05) or it is not significant. We are inclined to the alternative view, also commonly held, that the traditional 0.05 threshold is somewhat arbitrary (Cohen, 1994). The standard 0.05 threshold means that the chance of a data sample being a fluke is 1 in 20; whereas our 0.06 result simply means that the chance of our data sample being a statistical fluke is only slightly higher, at 1 in 17. We therefore argue that our replication results are weakly supportive of our original study. However, we also acknowledge it is possible that the effect we have observed in both the original study and this replication is on the margin of significance.

Alternately, one can argue that a p value around 0.05 is an encouragingly strong result, given that we are comparing student performance on just two explanation questions, one in week 3 and another in week 5. A more comprehensive test would involve asking several explanation questions in each of weeks 3 and 5.

2.2.1 Reflections

The results in Tables 4 and 5 support our suspicion that the earlier null result (i.e. Tables 2 and 3) may be due to how we taught the class. That is, even though the students may have rote learnt the swap code because of the emphasis we placed upon it in the lecture, the performance of students at explaining code in weeks 3 and 5 are consistent (albeit marginally consistent, at $p = 0.06$).

Even though the performance on the week 3 and week 5 explanation questions are (marginally) consistent, forty percent of students who answered well the week 3 explanation question did not answer well the week 5 explanation question. Such a backward step suggests (unsurprisingly) that some students who could reason correctly about the simpler code in week 3 could not transfer that reasoning to the week 5 problem containing an `if` statement.

Further to the point made in the previous paragraph, we wonder whether our use of a chi-square test is a pessimistic way of establishing the relationship between student performance on the week 3 and week 5 questions;

If you were asked to describe the purpose of the code below, a good answer would be "*It prints the smaller of the two values stored in the variables* `a` *and* `b`".

```
if (a < b):
    print a
else:
    print b
```

In one sentence that you should write in the empty box below, describe the purpose of the following code.

Do **NOT** give a line-by-line description of what the code does. Instead, tell us the purpose of the code, like the purpose given for the code in the above example (i.e. "*It prints the smaller of the two values stored in the variables* `a` *and* `b`").

Assume that the variables y1, y2 and y3 are all variables with integer values.

In each of the three boxes that contain sentences beginning with "`Code to swap the values …`", assume that appropriate code is provided instead of the box – do **NOT** write that code.

```
if (y1 < y2):
    Code to swap the values in y1
    and y2 goes here.
if (y2 < y3):
    Code to swap the values in y2
    and y3 goes here.
if (y1 < y2):
    Code to swap the values in y1
    and y2 goes here.
print y1
print y2
print y3
```

Sample answer: "it sorts the values so that y1 > y2 > y3"

Figure 3: A question from the week 5 test of Corney, Lister and Teague (2011).

Week 3, Explain a swap between two variables (see Figure 2)	Week 5, Successfully explained a sort of three variables (see Figure 3)
Wrong (n = 21)	33% (cf. 36%)
Right (n = 30)	60% (cf. 62%)
χ^2 test	p = 0.06 (cf. p = 0.03)

Table 4: Results from the replication at QUT, with comparative percentages shown in brackets from the original study.

Week 3, Explain a swap between two variables (see Figure 2)	Week 5, Successfully explained a sort of three variables (see Figure 3)	
	failure	Success
Wrong (n = 21)	14	7 (i.e. 33% of 21)
Right (n = 30)	12	18 (i.e. 60% of 30)

Table 5: The contingency table for calculating our chi-square value in Table 4.

for example, consider Table 5. A chi-square test focuses on consistency − whether most students answer both questions incorrectly (i.e. 14 in Table 5) or correctly (i.e. 18). It is not contrary to our argument, however, that some students would answer the week 3 question correctly, but the week 5 question incorrectly (i.e. 12). Our argument is merely that, in the absence of a pedagogical intervention, students who answer the week 3 question incorrectly will tend not to answer the week 5 question correctly (i.e. 7, which is 33% of 21)

2.2.2 Write the Swap but Explain the Code

To further test the idea that students had rote learnt the swap code in week 5, we looked at the n = 40 students who wrote the swap code correctly in week 5, and considered how well those students did on the week 3 and week 5 explanation tasks. The results are shown in Tables 6 and 7. These large differences in the two percentages (33% vs. 64%) do suggest that students who struggled to explain previously unseen code in week 3 tended to continue to struggle to explain previously unseen code in week 5. However, these percentages are not conclusive, as a χ^2 test produces a p value that is just over the traditional 0.05 threshold of statistical significance. We suspect that, with a slightly larger sample, the p value would meet the traditional 0.05 criterion.

Earlier, we suggested that the results in Tables 2 and 3 may be pedagogically encouraging, as those results may indicate that the problems students face are amenable to pedagogical intervention. In contrast, the results in Tables 6 and 7 are pedagogically discouraging − while students may have rote learnt how to swap the values of two variables, those students did not then manifest a strong ability to answer the week 5 explanation question.

3 Replication at UTS

In the replication at UTS, students were tested a little later in semester, at weeks 5 and 7. This was because the students were being taught an objects-early introduction to Java, so some of the concepts in the two tests were taught a little later in the semester.

In the week 5 test we used slightly different tracing questions to screen the students, but our questions also only involved assignment statements, and we do not regard these questions as being significantly different. After screening, 64 students remained.

We made one change in the replication that is arguably non-trivial. We changed one of the week 3 questions from the version shown in Figure 2 to the version shown in Figure 4.

Week 3, explain a swap − for the n=40 who wrote a correct swap in week 5	Week 5, explain a sort of three variables
Wrong (n = 15)	33% right
Right (n = 25)	64% right
χ^2 test	p = 0.06

Table 6: Performance of the 40 students who wrote a correct swap in week 5, on the week 3 and week 5 explanation problems.

Week 3, explain a swap − for the n=40 who wrote a correct swap in week 5	Week 5, explain a sort of three variables	
	failure	success
Wrong (n = 15)	10	5 (33% of 15)
Right (n = 25)	9	16 (64% of 25)

Table 7: The contingency table for calculating the chi-square value in Table 6.

The purpose of the following three lines of code is to swap the values in variables `a` and `b`, for any set of possible initial integer values stored in those variables:

```
c = a;
a = b;
b = c;
```

In one sentence that you should write in the box below, describe the purpose of the following three lines of code, for any set of possible initial integer values stored in those variables:

```
j = i;
i = k;
k = j;
```

Sample answer: "it swaps the values in i and k"

Figure 4: The modified question used in the replication at UTS. The original form of the question is in Figure 2.

3.1 Writing a Swap

Table 8 summarises our results from this part of the replication, where we consider student performance on explaining the swap code at week 5 and writing swap code at week 7. These results in Table 8 do not support our original results. As for the QUT replication, we suspect this null result is due to the week 5 test being followed by a lengthy discussion of the swap code by the lecturer (i.e. co-author Lister).

Week 5, Explain a swap between two variables (see Figure 4)	Week 7, Successfully wrote an equivalent swap between two variables
Wrong (n = 11)	82% (cf. 57%)
Right (n = 30)	79% (cf. 92%)
$\chi 2$ test	p = 0.8 (cf. p = 0.001)

Table 8: Results from the replication at UTS, with comparative percentages shown in brackets from the original study by Corney, Lister and Teague (2011).

3.2 Explaining a Sort of Three Variables

In our original study, we reported a statistically significant result for student performance on explaining swapping in week 3 and explaining the sorting of three variables in week 5. The results from our replication are shown in Tables 9 and 10. Our results emphatically confirm the results of the original study. As with the equivalent results from the replication at QUT, these results support our suspicion that the earlier null result (i.e. Table 8) is due to the lengthy lecture discussion about the swap code that followed the week 5 test.

Week 5, Explain a swap between two variables (see Figure 4)	Week 7, Successfully explained a sort of three variables (see Figure 3)
Wrong (n = 11)	27% (cf. 36%)
Right (n = 53)	91% (cf. 62%)
χ^2 test	p < 0.001

Table 9: Results from the replication at UTS, with comparative percentages shown in brackets from our original study (Corney, Lister and Teague, 2011).

Week 5, explain a swap between two variables (see Figure 6)	Week 7, explain a sort of three variables (see Figure 5)	
	failure	success
Wrong (n = 11)	8	3 (i.e. 27% of 11)
Right (n = 53)	5	48 (i.e. 91% of 53)

Table 10: The contingency table for calculating the chi-square value in Table 9.

4 Reflection: Ambiguity in Natural Language

A common concern among academics about "Explain in Plain English" questions is the possibility of ambiguity in student responses (e.g. Simon and Snowdon, 2011).

We found little ambiguity in our student responses – most answers were clearly right or wrong. For example, for the question on swapping values of two variables shown in Figure 2, some student responses that we judged as correct are:

swap b and c
swap contents of b and c using a as temp
swap values of c and b; leaving original value of b in a

Figure 5 shows some wrong answers given by students for this question. Most of these answers are clearly wrong.

For the question on sorting the values of three variables shown in Figure 3, some student responses that we judged as correct are:

orders in descending
places in descending
prints in order of highest to lowest
reorders in descending
sorts from largest to smallest
sorts in descending
swaps into descending

Figure 6 shows some wrong answers given by students for this question. Again, most of these answers are clearly wrong. Our experience is that grading student responses to explain in plain English questions is straightforward – arguably more straightforward than reading the confused code that students often write in exams.

1) a b and c have the same value
2) assigns a to b, b to c, c to b. No overwriting
3) b overwrites a; c overwrites b; then a overwrites c. b ends up in c
4) in the end, a will equal c, and c will equal a, both a and c hold same values
5) replaces c with b
6) sets a b and c to the value of b
7) swap values in b and a
8) to change every variable's value to that of b

Figure 5: A selection of wrong answers given by students at QUT for the code that swaps the values between two variables (see Figure 2).

1) assigns y3 the smallest value, y2 winds up with the default value
2) determines if a value is lower than another, then prints them all
3) if variables are smaller it will swap them to be larger at the end
4) printing, swapping y1 and y3 if y1 smaller
5) prints larger of 2 variables
6) prints largest value if one variable is smaller than the other
7) removes lowest and replaces with a value higher than it originally had
8) swap y1 and y3
9) swaps and prints
10) swaps codes for smaller value then print
11) swaps values y1 and y3, but y2 remains the same

Figure 6: A selection of wrong answers given by students at QUT for the Figure 3 code that sorts three variables.

5 Extension: A Third in-Class Test

At QUT, we went beyond the original study (Corney, Lister and Teague, 2011), but in a fashion very much in the same style as the original study, by conducting a third test in week 7 (i.e. mid-semester). In this week 7 test, we asked the students to write code to sort the values in an array with three elements. The code they needed to write is the same, algorithmically, as code they were asked to explain in week 5 (see Figure 3). However, the code they had to write in week 7 was not identical to the week 5 code, for the following reasons:

- Whereas the code in week 5 used three separate variables, the code in week 7 used a list of three elements.
- Students were required to write the actual assignment statements to swap values among the variables.

There were 48 students who did both the week 5 and week 7 tests. The results for all 48 students are shown in Table 11. It is not surprising that a low percentage (12%) of students who could not explain the code in week 5 could not also write similar code in week 7. More surprising was that only 35% of students who could explain the code in week 5 could write similar code in week 7. Once again, our p = 0.06 is just above the traditional 0.05 threshold for statistical significance, but as before we are inclined to believe that our results are weakly consistent with our original study, without meeting the traditional 0.05 threshold.

Week 5, explain a sort of three variables (see Figure 5)	Week 7, successfully wrote a correct sort of an array with 3 elements
Wrong (n = 25)	12%
Right (n = 23)	35%
χ^2 test	p = 0.06

Table 11: Results from week 7 test at QUT.

6 Replication: End of Semester Exam

In our original study, we reported a statistically significant relationship between student performance on their in-class tests and a code writing task in the final exam. In this section, we report our replication, again carried out at QUT.

In this replication, the code writing question in our end of semester exam was not the same as the question used in our original study. Our question in the replication required students to write code to move the elements of an array one place to the left, wrapping the leftmost element around to the rightmost position. One possible solution to this question is shown in Figure 7.

We screened students, using two tracing questions from the week 7 test. Both of these screening questions required students to trace iterative code operating on an array. If a student answered correctly at least one of those two questions, the student was judged as having demonstrated (as early as week 7) an understanding of the semantics of loops and lists. Since tracing iterative code is an error prone activity, especially as early as week 7, we felt that success on one question was sufficient evidence of understanding. Furthermore, tracing code with 50% accuracy is consistent with Lister's (2011) definition of the pre-operational stage in the novice programmer.

```
temp = x[0]

for i in range(0, len(x)-1, 1)
  x[i] = x[i+1]

x[len(x)-1] = temp
```

Figure 7: A solution, in Python, to the code writing question in the final exam.

When writing the solution to the problem in Figure 7, a student must provide a suitable assignment for the loop body, either `x[i] = x[i+1]` as shown in Figure 7, or `x[i-1] = x[i]`. We feel that students who failed to provide such an assignment statement demonstrated a profound misunderstanding of the question (perhaps due to English being their second language), so we also eliminated from our analysis any student who did not provide one of those two suitable assignment statements. The screening left us with a sample of 40 students.

Since this paper has emphasised the concept of swapping, our analysis of this exam question focuses upon the swapping component in this final exam question, especially the first and last lines as shown in Figure 7. The first line saves the leftmost element to a temporary variable, and the fourth line copies that temporary value back into the array.

(We note in passing that few students in the class gave a completely correct solution to this code writing problem. The most common errors in near-correct solutions were off-by-one errors in the loop. Often, the values through which the control loop variable "i" would iterate were appropriate, in isolation, and so was the assignment statement in the body of the loop. However, those two lines, in combination, were often not compatible.)

Table 12 breaks down the performance of students on this code writing task from Figure 7, according to whether the students were able to explain similar code in the week 7 test. Among students who could not explain that code in week 7, only 42% correctly handled the end element in the final exam, compared to 86% of students who did explain that code in week 7. A χ^2 test produces a statistically significant p value. Our result is therefore strongly supportive of our original findings.

Week 7, explained a shift (see code in Figure 3)	End of semester exam, write code to shift elements in an array (see Figure 7), correct treatment of the end element in lines 1 and 4
Wrong (n = 26)	42%
Right (n = 14)	86%
χ^2 test	p < 0.01

Table 12: Relative performance on the explanation task in week 7 and writing similar code in the final exam, at institution A (n=40).

7 Conclusion

Our empirical results support our original findings, with the following caveats.

In replications at both of our institutions, we did not find a relationship between students being able to explain swap code and being able to write similar code two weeks later. We believe this failure was because our teachers talked about the swap code between the two tests. In general, we think the relationship between explaining code and writing code found in our original study will only occur when there is not a pedagogical intervention between the two tests.

Some of our results were just outside the traditional 0.05 boundary of statistical significance, at $p = 0.06$. How readers will regard those results depends upon their view of the traditional 0.05 boundary. Some readers will maintain that a result is either significant (i.e. $p < 0.05$) or it is not significant. As we have argued earlier in the paper, we are inclined to the alternative view, which we believe is more statistically sophisticated, that the standard 0.05 threshold means that the chance of a data sample being a fluke is 1 in 20; whereas our 0.06 result simply means that the chance of our data sample being a statistical fluke is only slightly higher, at 1 in 17. We therefore argue that those replication results with $p = 0.06$ are weakly supportive of our original study, while acknowledging that our results do not meet the traditional $p = 0.05$ criterion. However, it is also possible that the effects we have reported in both the original study and in these replications are on the margin of significance. Further replication work, at other institutions, is warranted. Especially interesting would be further replication work that uses more than a single explanation question in each of the two weeks under comparison, as using only a single explanation question in each week may be the source of the statistical uncertainty.

One of our empirical results strongly supports our earlier findings, without the need for any caveats – we found that students who could not demonstrate an ability to explain a piece of code in week 7 of semester tended to do more poorly at attempting to write similar code at the end of semester.

Overall, this replication study and its minor extensions has increased our confidence in the conclusions we drew in the original study – the problems some students face in learning to program are not due to the more complex programming constructs they are taught in the second half of semester, but instead begin in the first half of semester.

References

Cohen, J. (1994) The Earth is Round ($p < .05$) *American Psychologist*, 49(12). pp 997-1003.

Corney, M., Lister, R., and Teague, D. (2011) *Early Relational Reasoning and the Novice Programmer: Swapping as the "Hello World" of Relational Reasoning*. Thirteenth Australasian Computing Education Conference (ACE 2011), Perth, Australia, January 2011. pp. 95-104.

Lister, R., Fidge C. and Teague, D. (2009) *Further Evidence of a Relationship between Explaining, Tracing and Writing Skills in Introductory Programming*. Fourteenth Annual Conference on Innovation and Technology in Computer Science Education, Paris, France. pp. 161-165

Lister, R., Clear, T., Simon, Bouvier. D., Carter, P., Eckerdal, A., Jackova, J., Lopez, M., McCartney, R., Robbins, P., Seppala, O., and Thompson, E. (2010) Naturally occurring data as research instrument: analyzing examination responses to study the novice programmer. *SIGCSE Bull*. **41**, 4 (January), 156-173.

Lister, R. (2011). *Concrete and Other Neo-Piagetian Forms of Reasoning in the Novice Programmer*. Thirteenth Australasian Computing Education Conference (ACE 2011), Perth, Australia. CRPIT, 114. John Hamer and Michael de Raadt Eds., ACS. 9-18.

Lopez, M., Whalley, J., Robbins, P., and Lister, R. (2008) *Relationships between reading, tracing and writing skills in introductory programming*. Fourth International Workshop on Computing Education Research, Sydney, Australia, 101–112.

Simon and Snowdon, S. (2011) *Explaining program code: giving students the answer helps – but only just*. Seventh International Computing Education Research Workshop (ICER), Providence, Rhode Island, pp. 93-99.

Venables, A., Tan, G. and Lister, R. (2009) *A Closer Look at Tracing, Explaining and Code Writing Skills in the Novice Programmer*. Fifth International Computing Education Research Workshop (ICER), Berkeley, CA. pp. 117-128.